

AD-A122 647

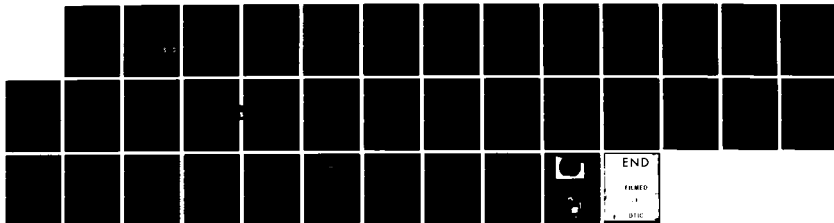
A DISCUSSION OF HARDWARE IMPLEMENTATION AND FABRICATION  
FOR AN AUTOMATIC. (U) MARYLAND UNIV COLLEGE PARK  
COMPUTER SCIENCE CENTER T WILLETT ET AL. 29 JUL 77  
DRAG53-76-C-0138

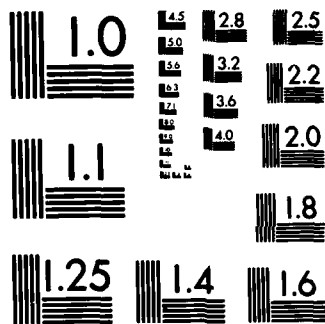
1/1

UNCLASSIFIED

F/G 19/5

NL





MICROCOPY RESOLUTION TEST CHART  
NATIONAL BUREAU OF STANDARDS-1963-A

AD A122647

**A DISCUSSION OF HARDWARE IMPLEMENTATION AND FABRICATION  
FOR AN AUTOMATIC TARGET CUEING SYSTEM**

July 29, 1977

This is the fifth quarterly status report on a program for Recognition Technology for a Smart Sensor, conducted by Westinghouse for U. Md. under Contract DAAG 53-76-C-0138 with the U. S. Army Electronics Command, Night Vision Laboratory, Ft. Belvoir, Va. 22060

Prepared for

Computer Science Center  
University of Maryland  
College Park, Maryland 20742

DTIC  
ELECTE  
DEC 22 1982  
S B D

By

Westinghouse Defense and Electronic Systems Center  
Systems Development Division  
Baltimore, Maryland 21203

APPROVED FOR PUBLIC RELEASE  
DISTRIBUTION UNLIMITED

82 12 15 097

DTIC FILE COPY

## TABLE OF CONTENTS

### INTRODUCTION

#### 1.0 SYSTEM FLOW

##### 1.1 Algorithms

- 1.1.1 Median Filter
- 1.1.2 Gradient Operator
- 1.1.3 Non-Maximum Suppression
- 1.1.4 Gray Level Threshold Determination
- 1.1.5 Connected Components
- 1.1.6 Perimeter Match
- 1.1.7 Feature Extraction

#### 2.0 HARDWARE IMPLEMENTATION

##### 2.1 Connected Components

- 2.1.1 Line Delay with 100 Colors
- 2.1.2 Data Rates
- 2.1.3 Equivalence Statements
- 2.1.4 Feature Extractor
- 2.1.5 Coloring Operator
- 2.1.7 System Flow

#### 3.0 CHIP DEVELOPMENT

Accession For	
NTIS GRA&I	<input checked="checked" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A	



## INTRODUCTION

This is the fifth quarterly status report on a program for RECOGNITION TECHNOLOGY FOR A SMART SENSOR, being conducted by Westinghouse Systems Development Division for the Computer Science Center, University of Maryland. The program consists of three phases, as follows:

PHASE I      Task and Technology Review (3 months)

PHASE II     Algorithm Selection and Test (9 months)

PHASE III    Hardware Development (9 months)

This report covers the first three months of the Phase III effort. The report was prepared by Mr. Thomas Willett, Dr. Nathan Bluzer and Dr. Gerald Borsuk of Westinghouse. The Westinghouse program manager is Dr. Glenn E. Tisdale.

During the quarter, five meetings were held between members of the Maryland and Westinghouse teams. Mr. John Dehne and Dr. George Jones of NVL attended several of the meetings.

Westinghouse is concentrating on the hardware implementation and fabrication of the U. of MD algorithms for the focal plane and treating them as a system. This quarter continues the emphasis on implementation and fabrication. The work this quarter on more intelligent operators (image segmentation, feature extraction, and classification) in CCD technology continues. A progress report on a histogrammer chip fabrication for a feasibility demonstration is included.

## 1.0 SYSTEM FLOW

This section describes a preferred set of algorithms developed by Maryland which tentatively comprises the first portion of a cueing system. A system flow chart is shown in Figure 1-1. A description of data flow and storage requirements is included.

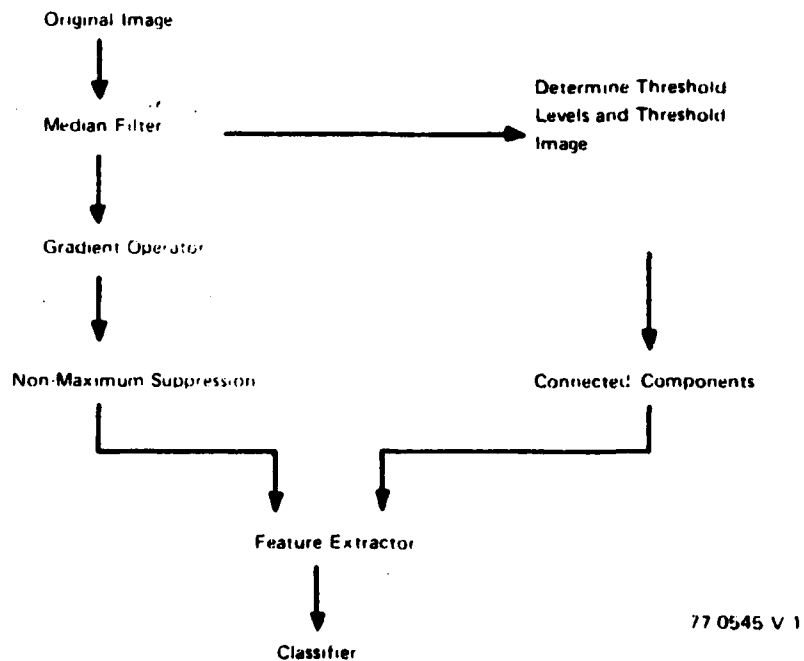


Figure 1-1. System Flow Chart

In general, the Median Filter acts to suppress noise. The Gradient Operator extracts edges which are then thinned by the Non-Maximum Suppression Algorithm. At the same time a set of gray levels is determined and the

filtered image is thresholded at each gray level. A Connected Components Algorithm partitions the thresholded image into object regions. A matching Algorithm correlates perimeter points formed independently by the Non-Maximum Suppression and Connected Components Algorithms and a score is obtained. This score and several other algorithms form a set of Classification Logic.

## 1.1 Algorithms

A short description of each algorithm follows.

### 1.1.1 Median Filter

This algorithm was described in the second quarterly report.

### 1.1.2 Gradient Operator

This algorithm was described in the second quarterly report.

### 1.1.3 Non-Maximum Suppression

This algorithm was described in the third quarterly report.

### 1.1.4 Gray Level Threshold Determination

U. Md. is evaluating several approaches to this determination; hardware implementation will be held in abeyance until completion.

### 1.1.5 Connected Components

This algorithm was described in the third quarterly report. Because the algorithm is the subject of the Hardware Implementation Section of this quarterly report, we repeat that description here. The purpose of the algorithm

is to segment the image data stream into smaller domains. Each small domain includes a single object in the image plane. This algorithm distinguishes between objects and isolates regions so that statistics for Classification Logic can be obtained.

• Assume that the original image has been thresholded and the result is in binary form with gray levels exceeding  $g_1$  shown as 1's in Figure 1-2.

1			1	1	1	1				1	1	
1			1	1	1		1				1	
1			1	1	1		1				1	1
1			1	1	1	1	1				1	
1			1	1	1	1	1	1	1		1	1
										1	1	1

Binary Image

77-0189-V-3

Figure 1-2a. Binary Image

A			B	B	B	B				D	D	
A			B	B	B		C				D	
1			1	1	1		1				1	1
1			1	1	1	1	1				1	
1			1	1	1	1	1	1	1		1	1
										1	1	1

77-0189-V-4

Figure 1-2b. Computations for Second Row

Two image lines are retained in memory so that each pixel can examine its neighbors to the left and right and above and below. No diagonal connections are permitted under this convention, and an adjacent (horizontal or vertical) pixel must be occupied in order to make a connection. No skips or gaps are allowed, and the computations start one pixel in from the edge. In Figure 1-2b, there are four distinct regions, A, B, C, and D. The only possible connection between regions B and C is through a diagonal, which is not allowed. Computations for the fourth row are seen in Figure 1-2c.

A			B	B	B	B				D	D	
A			B	B	B		C				D	
A			B	B	B		C				D	D
A			B	B	B	C	C				D	
I			I	I	I	I	I	I	I		I	I
										I	I	I

77-0189-V-5

Figure 1-2c. Computations for Fourth Row

Here, there is a connection between regions B and C and an equivalence statement,  $B = C$ , is carried along. At the end of the sixth row, there is another connection between C and D ( $C = D$ ) and all the regions are completed as seen in Figure 1-2d.

A			B	B	B	B				B	B	
A			B	B	B		B				B	B
A			B	B	B		B				B	
A			B	B	B		B				B	
A			B	B	B	B	B	B	B		B	B
									B	B	B	

77-0189-V-6

Figure 1-2d. Completed Image

The areas of A, B, C and D are computed by cumulating the number of pixels assigned to each. The perimeter is calculated by cumulating the number of pixels assigned to each region which are neighbors of zeros, i.e., the neighbors did not exceed the gray level threshold,  $g_1$ .

#### 1.1.6 Perimeter Match

This algorithm was described in the third quarterly report.

#### 1.1.7 Feature Extraction

Maryland has begun to designate features to be extracted from the Connected Components Algorithm for classification. A tentative list of features, including perimeter match, contains specified average gray level over the target, perimeter extent, area, and maximum height and width.

## 2.0 HARDWARE IMPLEMENTATION

### 2.1 Connected Components

The purpose of the Connected Components Algorithm is to segment an image frame into object regions; these object regions are potential shapes of interest and features are extracted from them for classification purposes. We assume that Time Delay Integration is part of focal plane signal processing which implies that the image comes to the cuer in the form of one line at a time, i.e., the pixels in one line arrive in parallel. The Connected Components Operator then moves along this line of pixels, with the previous line in memory, determining which pixels are part of a particular object region or if a new object region is starting. If we are to extract features from each object region, there must be a means for distinguishing between different object regions. One approach to the problem is to color each object region with a different color and then have a feature extractor operator assigned to each color. Where an object has several colors, the feature extractors corresponding to those colors accumulate their features, dump them in a scratch feature extractor to combine them, and reassign the result to one of the two feature extractors. This has been a description of the general approach; let us now proceed into the specific technical problems.

#### 2.1.1 Line Delay with 100 Colors

At the shift rates and number of shifts involved, we are conservatively estimating that ten analog levels can be maintained and distinguished per CCD site. To produce 100 colors, which seems to be enough, we could use ten CCD shift registers in parallel where one vertical column, ten sites high, is equivalent to one pixel of the image. One line of image memory would

be approximately 600 CCD sites long by 10 rows high. We are continuing with our conservative assumption of surface channel CCD's, even though Westinghouse is currently producing buried channel devices with transfer rates in excess of 7 megahertz. These peristaltic devices will reduce the size estimates by roughly a factor of ten. Figure 2-1 shows the one line delay without coding. The ten rows of shift registers could be reduced to two and still

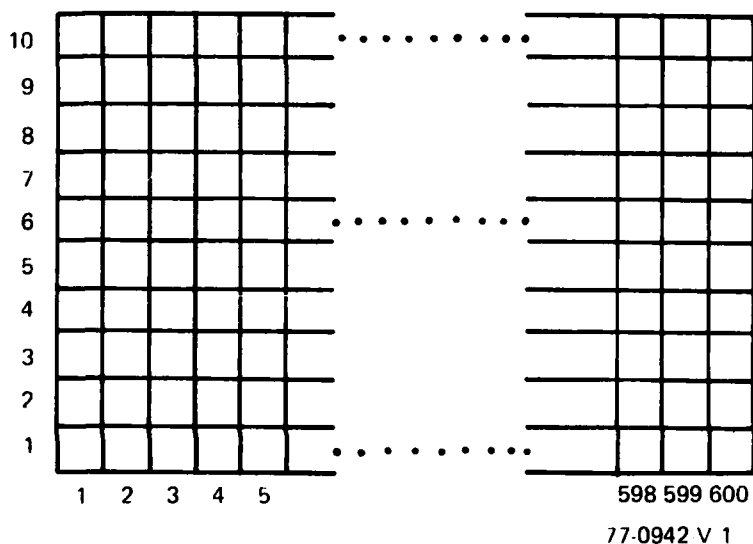
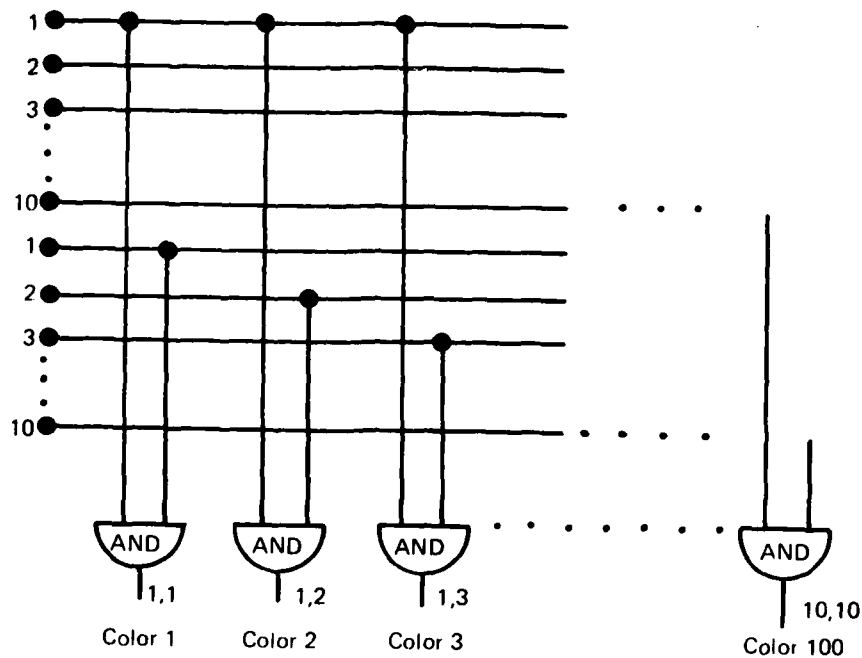


Fig. 2-1. One Image Line Delay Containing 100 Colors Without Coding

obtain 100 colors by organizing the decoder in the form of a Field Programmable Logic Array (FPLA). The color is quantized to ten levels and the highest bin containing a quantum of charge is identified (the identification procedure was described in the last quarterly) for each shift register. Then the quantum levels for the two registers are catenated and the leads connected to AND gates as shown in Figure 2-2. In summary, we have described two methods for constructing a line delay containing 100 colors.

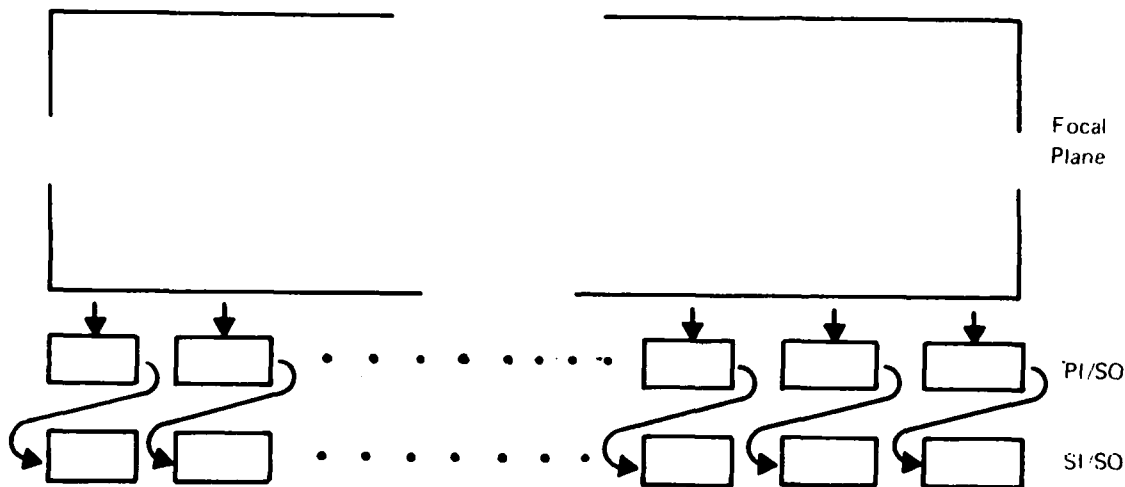


77-0942 V 11

Figure 2-2. One Image Line Delay with Coding

### 2.1.2 Data Rates

The desired data rate is 1 megapixel/sec. and to achieve this with surface channel CCD's with an assumed rate of 50-100 KHz requires multiple Connected Components Operators in the same manner as multiple Median Filter, Gradient, or Non Maximum Suppression Operators. That is, the line delay (now two lines deep) would be divided into a number of vertical columns and a Connected Component Algorithm performed within each column as shown in Figure 2-3. Since there are 20 columns, we would restrict each column to a particular 5 colors. Also, there would be 5 feature extractors, each corresponding to a particular color, per column. However, the object region must still be constructed across columns since an object region may be 100



77 0942 V 3

Figure 2-3. Partitioned Processing

pixels or 4 columns wide. One can begin to envision a hierarchy or branching structure of Connected Component Operators. The point, here, is an important one: with the primitive operators, we were able to break the image into columns because the operators were relatively independent between columns. That is not the case here, and it appears that one Connected Component Operator is desirable. This implies a buried channel CCD device as opposed to a surface channel one. A surface channel device has more dynamic range, perhaps 6 db, but dynamic range is not as important as speeds of 10 megapixels/sec.

### 2.1.3 Equivalence Statements

The problem occurs in a multi-colored object where the colors must be combined so that the object features are accumulated together. It was noted

Here, each feature extractor is associated with one and only one color. When features of an object region are being accumulated and it is discovered that part of the region has a different color, accumulation continues for both colors. But the contents of both feature extractors are dumped into a scratch feature extractor which accumulates both color features. The results are put in the dominant color feature extractor and the subordinate color feature extractor is zeroed. One way to perform this zeroing process is to wait until the individual components are completed and then combine colors. For example, consider the object region shown in Figure 2-4. Coloring the object from left

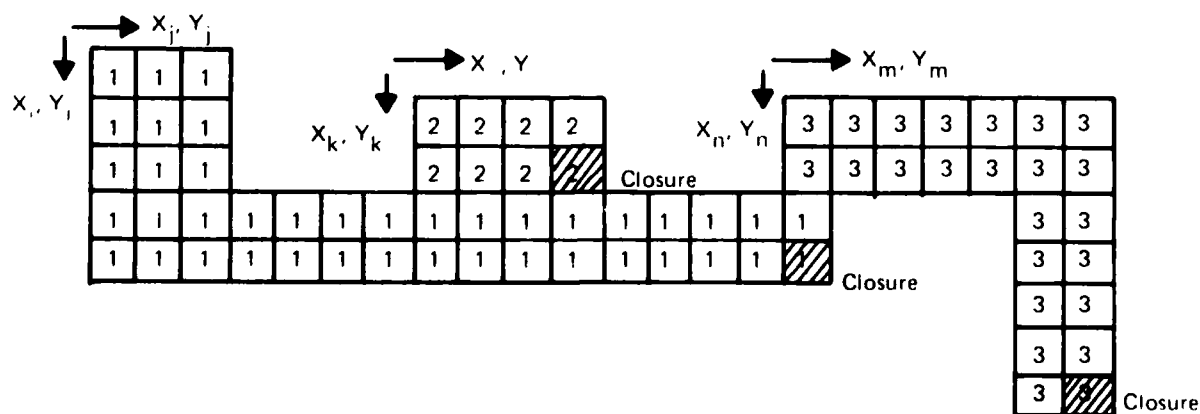
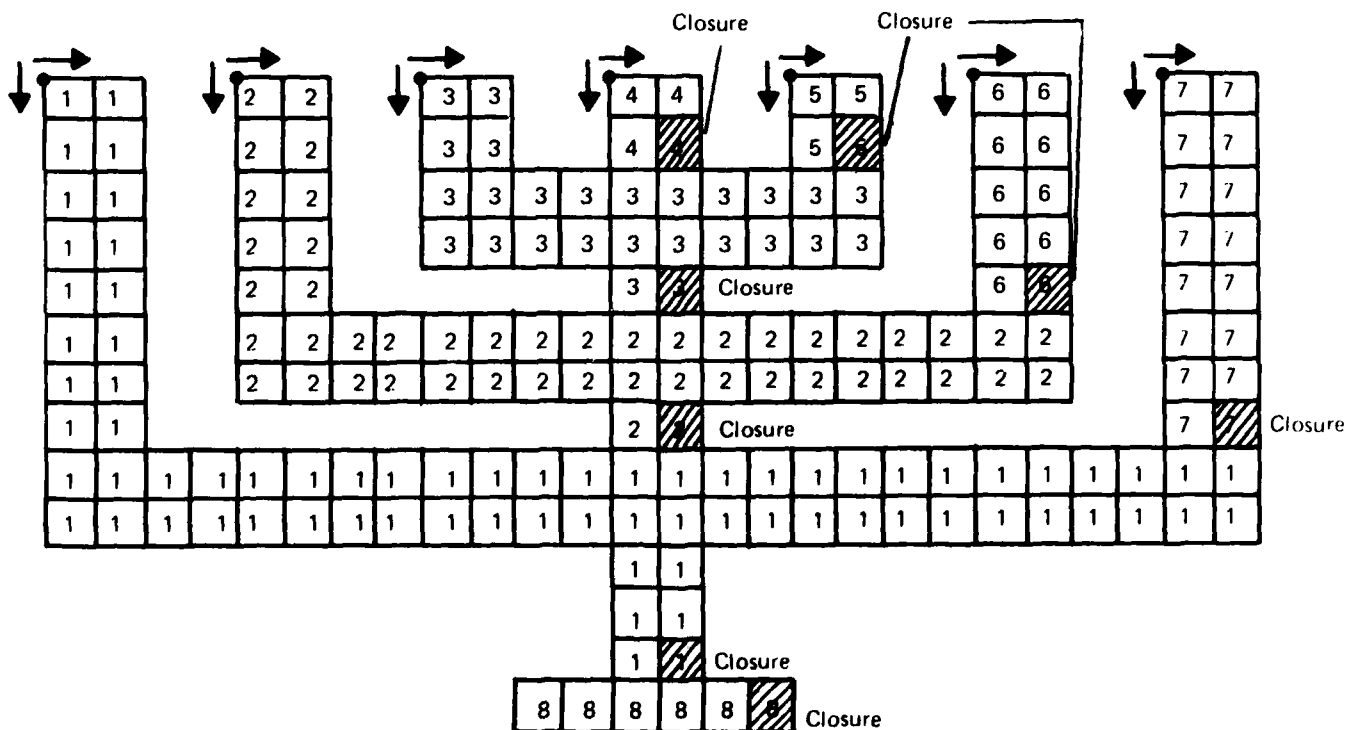


Figure 2-4. Arbitrary Object Region

to right and from top to bottom results in a pattern shown in Figure 2-4 assuming rows dominate columns. If two pointers were assigned to each color which traced the perimeter in both x and y to indicate, then the colors could be combined when each component was completed. These pointers would occupy the first four accumulator positions in the feature extractor. This is a method of dealing with the equivalence statements. If we attempt to execute equivalence statements as they arise in the figure, certain problems occur. For example, in Figure 2-4, the tail composed of 3's would never be combined with 2's after the first four rows. An alternative to this is to introduce a feature extractor for each column, analogous to the digital implementations, which increases the number of feature extractors to approximately 600, an increase of 6 times. So this implementation, also, should be avoided. Consider how this implementation would perform on the object region shown in Figure 2-5. Again, we are coloring from left to right and from top to bottom. Note that colors 4 and 5 are closed out at row 2, colors 3 and 6 are closed out at row 5, colors 2 and 7 are closed out at row 8, and the last rows close out colors 1 and 8. In the coloring scheme, we have arbitrarily decided that rows will dominate columns, so that at the intersection of colors 3 and 4, e.g., color 3 (the column color) will dominate. Further, we have designated the dominant and subordinate colors in the equivalence statements as row and column colors. The equivalence statements are shown in Figure 2-6. Recall, that when a color is closed, it is merged via an equivalence statement. It seems reasonable to try and close columns off as we move across the rows. If we set the object on end and colored it, the results would be as shown in Figure 2-7. Again, we invoke the equivalence rule of merging a closed out color into one which is not closed.



77-0942-V-5

Figure 2-5. Arbitrary Object Region

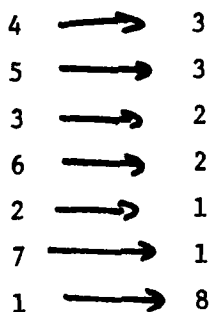


Figure 2-6. Equivalence Statements for Figure 2-5.

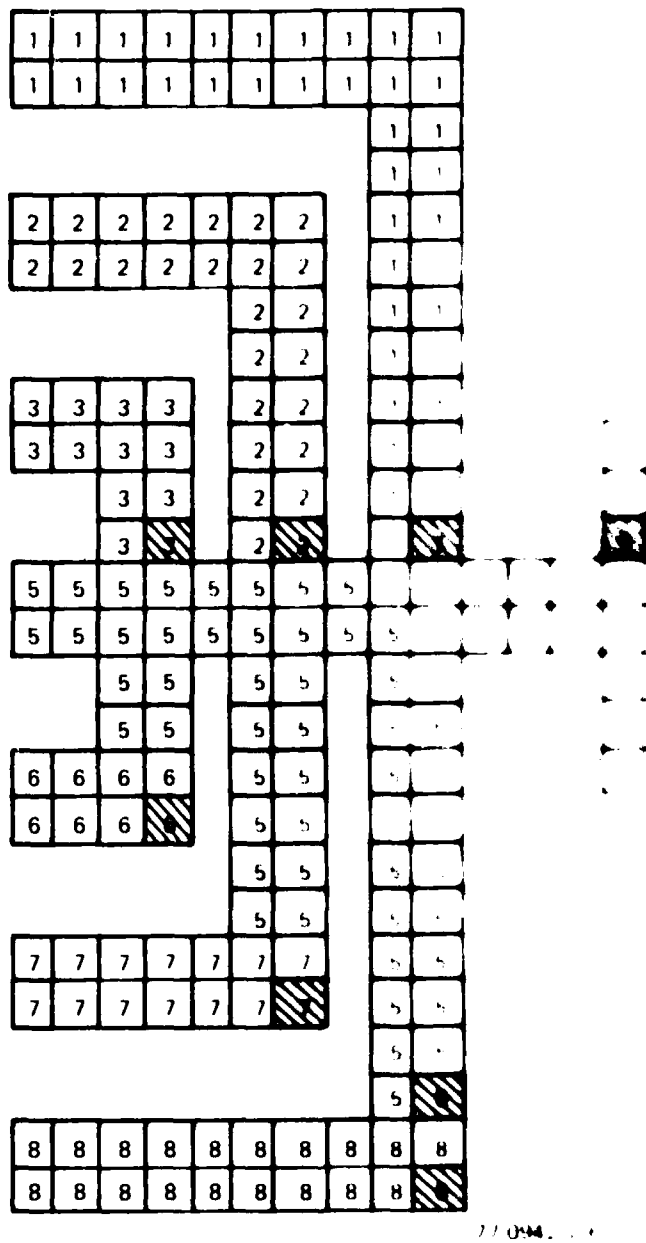


Figure 2-7. Candelabra on End

Thus the equivalence statements for the candelabra on end are shown in Figure 2-8.

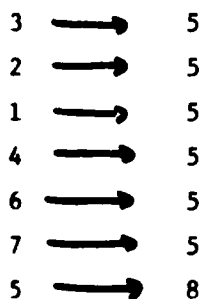
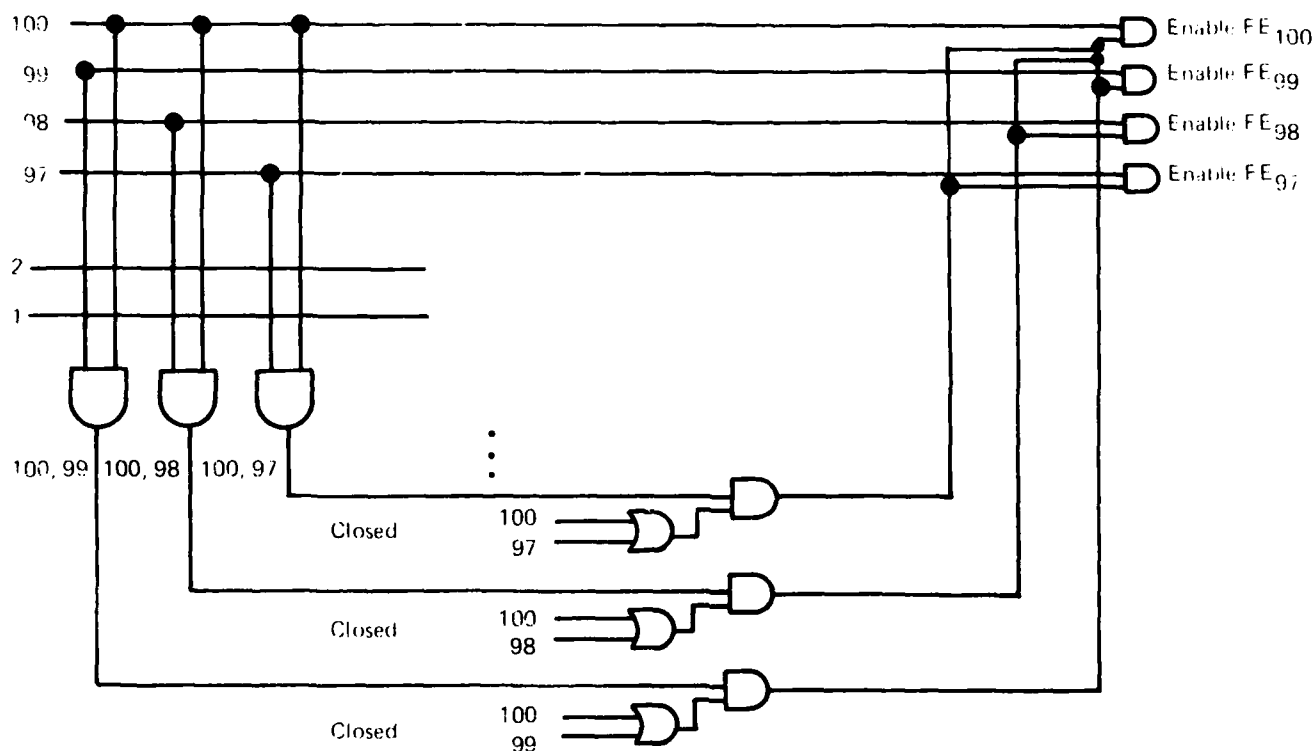


Figure 2-8. Equivalence Statements for Figure 2-7

When we are waiting until a color is closed out before invoking the equivalence statements, we are implying some sort of memory which holds the equivalence statement components. The memory is enabled when a color is closed out and the direction of the equivalent statement is indicated.

We need three signals to activate an equivalence statement: both components of the statement and a signal indicating that one of the components has been closed out. This signal also determines the direction of the equivalent signal. Again, the 100 colors are fed to an FPLA - like structure to indicate the horizontal and vertical connections as shown in Figure 2-9. Note that all color combinations are allowed. The control signals are not sent to the appropriate Feature Extractors until one or the other of the colors is closed out as indicated by the OR gate; this output is then ANDed with the combination output to indicate the presence of all three required signals. Once the color closure signal is received, the circuit is reset.

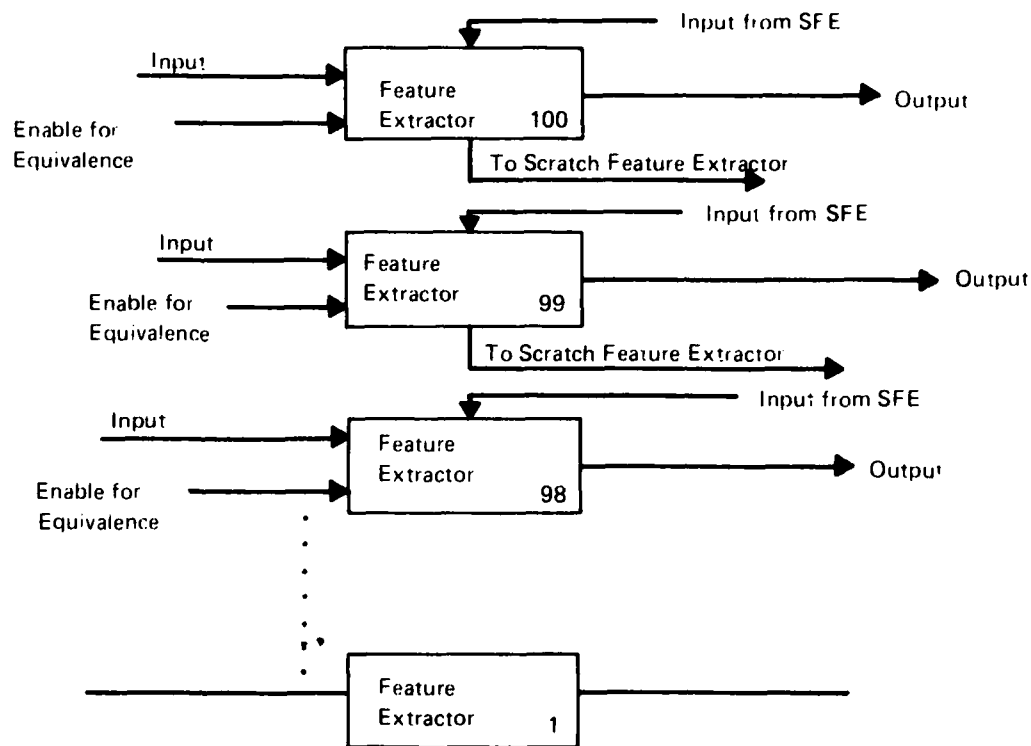


77 0942 V 7

Figure 2-9. Circuit for Directing Equivalent Statements.

#### 2.1.4 Feature Extractor

There is a Feature Extractor corresponding to each color as shown in Figure 2-10. The signals shown in Figure 2-9 enable the particular Feature Extractors for the equivalence computations. They also direct which Feature Extractor will receive the contents of the scratch Feature Extractor, i.e., that color which is not closed out. We have organized Figure 2-9 so that the number of inputs to the Feature Extractors is minimized. Each Feature Extractor is visualized as a many-channeled, large holding well which follows along the lines of the sorter. (See Fourth Quarterly Report)



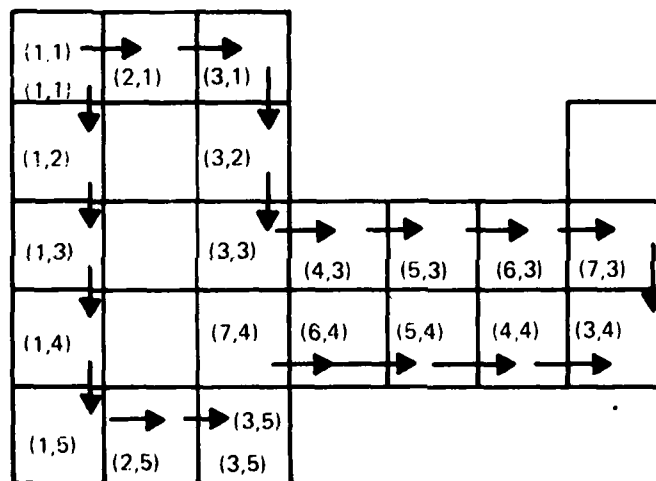
77-0942-V 8

Figure 2-10. Organization of Feature Extractors

Each channel would correspond to a particular feature and since the features are cumulative, they would simply add in the scratch feature extractor. A discussion and implementation of the features as derived thus far by the University of Maryland was contained in the Fourth Quarterly Report.

A new feature which Westinghouse has added to support the implementation of Connected Components is the use of pointers which indicate when a color is closed out or completed. The pointers essentially trace the outline of

the color component in two directions, starting from the same point, and the color component is completed when the pointers meet. The two pointers,  $(x_i, y_i)$  and  $(x_j, y_j)$  each have an x and y component which add positively and negatively as the pointers trace the component outline. These pointers are checked at each pixel for a close-out condition. The pointers may be expressed in terms of local Component Coordinates, rather than the coordinates within the frame. When the difference between the two pointers is zero, the component is closed out. Let us examine Figure 2-11 as an example of the use of pointers.



77-0942-V-9

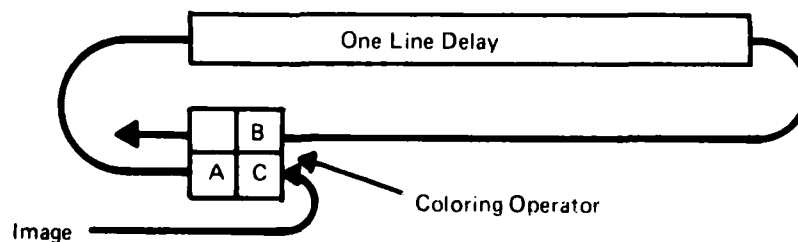
Figure 2-11. Pointer Illustration

At the end of row 3, the two pointers are standing at (1,3) and (7,3). Beginning row 4, the first point is updated to (1,4), however the problem is to bring the second pointer back to the left. Since we are in the fourth row and perimeter points are being encountered, the pointer can be upgraded to (7,4) and then the number of perimeter points in the x direction subtracted. So, as we move across the 4th row from left to right, the point becomes,

successively, (7,4), (6,4), (5,4), (4,4), and (3,4). This method of subtraction obviates the necessity for an extra line or two of memory. Before going any further into the pointer logic, let us first consider the logic for the coloring operator.

#### 2.1.5 Coloring Operator

We assume that the Coloring Operator processes a binary image, i.e., each pixel contains either a one (1) or a zero (0). The binary data stream will enter the Coloring Operator and emerge transformed into different colors or signal levels for different shapes. Read out of the binary picture will progress one horizontal line at a time starting with the top line and progressing downward. Each horizontal line will be read out from left to right. Since the image data is read out serially, the Coloring Operator is a local operator. The coloring of each non-zero element in the image plane will be done as shown in Figure 2-12.



77 0942 V-10

Figure 2-12. Data Flow Through Coloring Operator

The Coloring Operator is a transformation from a binary picture to a colored one by a mapping T

$$T(A, B, M): C \rightarrow C^1$$

where  $C^1$  is the color of the transformed pixel C, the variables A and B represent nearest neighbors of C, and M represents an available color. The relative locations of pixels A, B, and C in the image plane are shown in Figure 2-13.

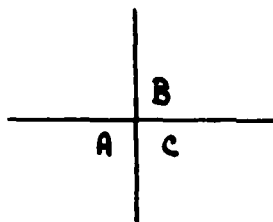


Figure 2-13. Relative Location of A, B, C

We define the coloring window as always containing these three elements. Elements A and B are nearest neighbors of C and have already been processed by the Operator. Element B is located one horizontal line above elements C and A. Element C is being painted by the Coloring Operator according to the following rule:

$$\text{For } C \neq 0 \quad C^1 = \begin{cases} A & \text{if } A \neq 0, B \neq 0 \\ B & \text{if } A = 0, B \neq 0 \\ M & \text{if } A = 0, B = 0 \end{cases}$$

When adjacent elements have different colors, the element being pointed assumes the color of the nearest neighbor in the same line (rows dominate). Whenever elements A and B are zero and element C is not zero, element  $C^1$  is given a new color. Implementation of the Coloring Operator was described in the

FourthQuarterly Report. Up to now, we have been reviewing the work already described in theFourth Quarterly Report; we want to expand the capability of the Coloring Operator to include multicolor detection (detect equivalent statements) and pointers for detecting component completion (color closeout).

Multi-colored object regions occur when pixels A and B are both non-zero, not equal to each other, and C is non-zero, i.e.,

For  $C \neq 0$ ,  $A \neq B \neq 0$

When this is true, A and B are non-destructively read out from the Coloring Operator and sent to the logic array shown in Figure 2-9.

The pointer logic is more elaborate since we shall be concerned about perimeter pixels and maintaining count. We want to avoid introducing another line of delay, so the Coloring Operator is expanded horizontally by adding pixels D, E, and F as shown in Figure 2-14.

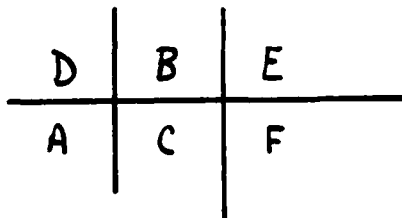
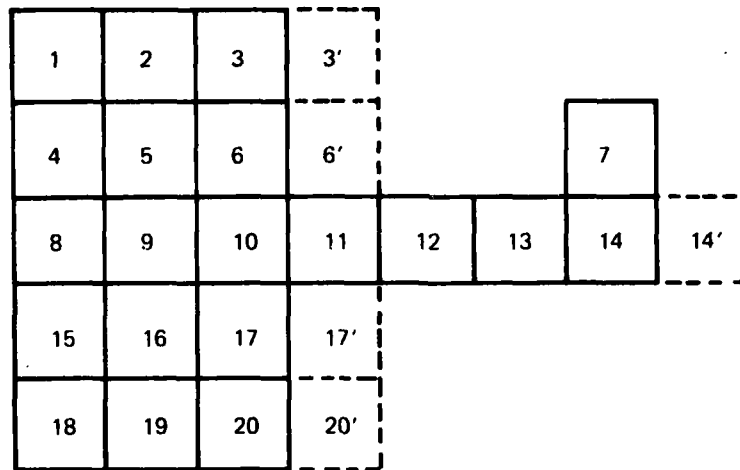


Figure 2-14. Relative Locations of A,B,C,D,E,F

Now, let us examine Figure 2-11 with the moving window just shown. We have shown the object region again in Figure 2-15 with each pixel numbered to correspond to the truth table shown in Figure 2-16. The problem with this



77-0942-V-2

Figure 2-15 . Pixels Numbered

double set of pointers is that at pixel 14 there is no way to bring the second pointer back along the single row of object region (pixels 8 - 14) without another layer of logic or another line of delay. See the condition of point  $X_A, Y_A$  at pixels 15-17.

Another approach to the problem of color close-out is to use only two pointers  $(x_1, y_1)$  for each color where  $x_1$  moves horizontally and  $y_1$  moves vertically. Each is clocked according to the row and column clocks respectively. The truth table for this simpler arrangement is shown in Figure 2-17. Note that pixels E and F have been eliminated for this implementation. From constructing the Truth Table of Figure 2-17, we see that the implementation reduces again to a simpler form. Every time the image is cycled through an entire line, the column clock goes from  $m$  to  $m + 600$  (assuming each image line is 600 pixels long). If say column  $m$  is the last column to contain color  $K$  and the column clock goes to  $m + 600$  without encountering another pixel containing color  $K$ , then color  $K$  may be closed out and the equivalence statements containing color  $K$  may be enacted.

Pixel No.	A	B	C	D	E	F	X <sub>D</sub>	Y <sub>D</sub>	X <sub>A</sub>	Y <sub>A</sub>	Col	Row
1	0	0	1	0	0	1	1	1	1	1	n	m
2	1	0	1	0	1	1	1	1	2	1	n+1	m
3	1	0	1	0	0	0	1	1	3	1	n+2	m
4	0	1	1	0	1	1	1	2	3	1	n	m+1
5	1	1	1	1	1	1	1	2	3	1	n+1	m+1
6	1	1	1	1	0	0	1	2	3	2	n+2	m+1
7	Different Color											
8	0	1	1	0	1	1	1	3	3	2	n	m+2
9	1	1	1	1	1	1	1	3	3	2	n+1	m+2
10	1	1	1	1	0	1	1	3	3	3	n+2	m+2
11	1	0	1	1	0	1	1	3	4	3	n+3	m+2
12	1	0	1	1	0	1	1	3	5	3	n+4	m+2
13	1	0	1	1	1	1	1	3	6	3	n+5	m+2
14	1	0	1	0	0	0	1	3	7	3	n+6	m+2
15	0	1	1	0	1	1	1	4	7	3	n	m+3
16	1	1	1	1	1	1	1	4	7	3	n+1	m+3
17	1	1	1	1	0	0						

77-0942-T-12

Figure 2-16. Truth Table

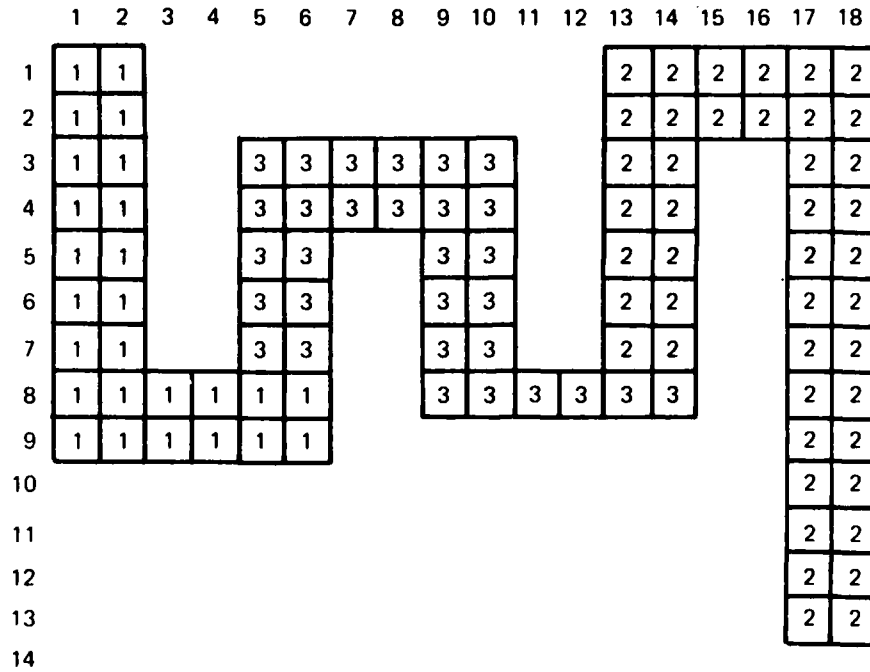
Pixel No.	A	B	C	D	X	Y	Col. Clock	Row Clock
1	0	0	1	0	1	1	m	n
2	1	0	1	0	2	1	m+1	n
3	1	0	1	0	3	1	m+2	n
3'	1	0	0	0	3	1	m+3	n
4	0	1	1	1	3	2	m	n+1
5	1	1	1	1	3	2	m+1	n+1
6	1	1	1	1	3	2	m+2	n+1
6'	1	0	0	1	3	2	m+3	n+1
7	A Different Color							
8	0	1	1	0	3	3	m	n+2
9	1	1	1	1	3	3	m+1	n+2
10	1	1	1	1	3	3	m+2	n+2
11	1	0	1	1	4	3	m+3	n+2
12	1	0	1	0	5	3	m+4	n+2
13	1	0	1	0	6	3	m+5	n+2
14	1	0	1	0	7	3	m+6	n+2
14'	1	0	0	0	7	3	m+7	n+2
15	0	1	1	0	7	4	m	n+3
16	1	1	1	1	7	4	m+1	n+3
17	1	1	1	1	7	4	m+2	n+3
17'	1	1	0	1	7	4	m+3	n+3
18	0	1	1	0	7	5	m	n+4
19	1	1	1	1	7	5	m+1	n+4
20	1	1	1	1	7	5	m+2	n+4
20'	1	1	0	1	7	5	m+3	n+4

77-0942-T-13

Figure 1-17. Truth Table, Second Implementation

A circulating CCD shift register seems directly applicable to this implementation. The register contains all the colors in the last line of the image. In order to continue a color (and not close it out), there must be a vertical connection somewhere along the next image line. One possibility is to construct a "timer" for each color such that a vertical connection would reset the timer. If the timer were allowed to reach 600, it would enable the equivalence statement logic and close out the color. The timer could take the form of one of the channels of the large holding well found in the feature extractor for each color. At the rate that the pixels are shifted in the system, a quantum of charge would be entered into the timer channel; the accumulated charge would be non-destructively read out and compared to say 600. If the accumulated charge were larger than an amount equivalent to 600 shifts, the color would be closed out. If there were a vertical connection, the contents of the timer channel would be reset (if  $\leq 600$ ) to zero, and the accumulation started again for that color. Note that each color has its own timer and each is updated at every clock pulse.

2.1.6 There is an additional piece of logic which must be discussed before the system block diagram of the Connected Components implementation is shown. In discussions between NVL, the University of Maryland, and Westinghouse, certain difficult shapes representing object regions have been discovered. At various times, we have considered the techniques of closing out colors and recoloring the last line separately, but it seems that when they are combined the algorithm proceeds smoothly. Consider the object region of Figure 2-18 as one of the more difficult shapes



77-0942-V-14

Figure 2-18a. Object Region.

At line number 8, vertical connections are found between colors 1 and 3 and 2 and 3. At line 9, and column 15, color 3 is closed out and the equivalence statements 3 - 1 and 3 - 2 are executed. At line 10 and column 7, color 3 is closed out but there is no equivalence statement to be executed. Color 2 is closed out at line 14 and column 19, but there is no equivalence statement between colors 1 and 2 so the object region is treated as two regions instead of one. A different approach is shown in Figure 2-18b.

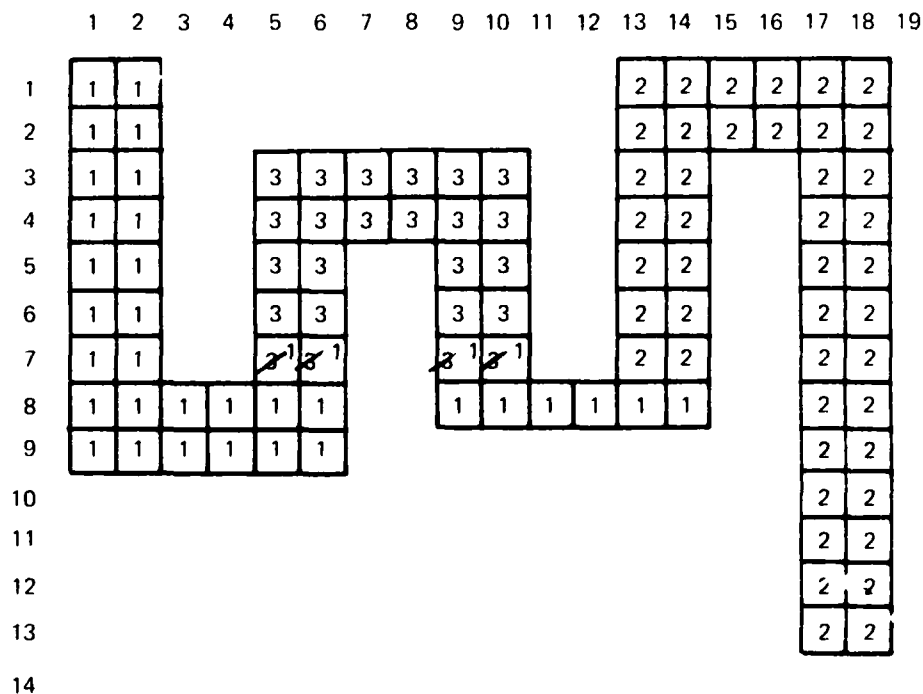


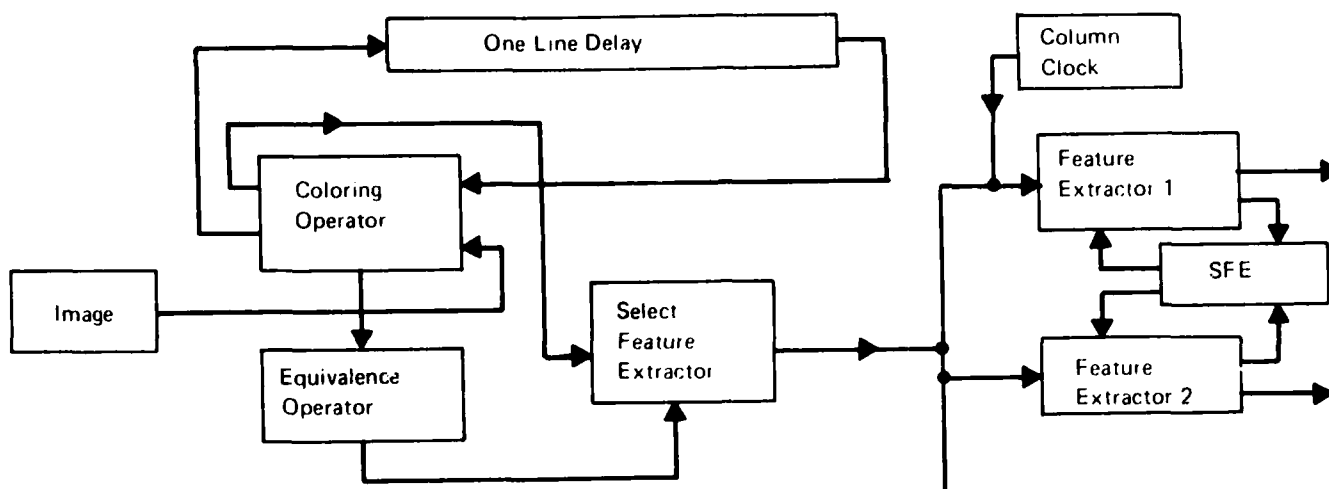
Figure 2-18b.

77-0942 V 15

Here, the last row is recolored when a vertical connection is discovered and the new color is substituted for the old along the line. This allows colors 1 and 2 to eventually meet forming the required equivalence statement and allowing the object to be considered as one region instead of two.

#### 2.1.7 System Flow

In this section on connected components, we started out by briefly discussing the overall system flow and launched into five (5) separate discussions on parts of the system. Now, we want to return to a system description of the Connected Components Algorithm.



77-0942 V-16

Figure 2-18. System Block Diagram

Read from the top of the image to the bottom and from left to right. The one hundred (100) color delay line is represented by twenty (20) SI/SO CCD delay lines which are coded to obtain 100 colors and obviate transfer efficiency problems. There are 20 levels of color comparisons for horizontal and vertical connections in the Coloring Operator. The Equivalence Operator notes horizontal and vertical connections between different colors and the close-out of a color, thus activating the equivalence logic. The column clock is actually fed to all the Feature Extractors and they indicate when a color is closed out. The device which selects the appropriate Feature Extractor is a decoder which was shown in Figure 2-2.

### 3.0 CHIP DEVELOPMENT

The Smart Sensor Project is scheduled to last for 21 months with a key circuit selected at the one-year mark and constructed in the last nine months. We want to select a circuit which is common to as many of the algorithms as possible. Figure 3-1 shows the algorithms and the functions which are required by each algorithm. A perusal shows that the sorter function occurs in four out of the five algorithms and is the one we selected.

<u>ALGORITHM</u>	<u>FUNCTION</u>
GRADIENT OPERATOR	ABSOLUTE DIFFERENCE COMPARISON
MEDIAN FILTER	QUANTIZER SORTER
NON-MAXIMUM SUPPRESSION	QUANTIZER SORTER ABSOLUTE DIFFERENCE
CONNECTED COMPONENTS	QUANTIZER SORTER COLOR LINK
HISTOGRAM	QUANTIZER SORTER

Figure 3-1: Algorithm Implementation by CCD Function

Several versions of the sorter were put in production runs at the Westinghouse Advanced Technology Laboratory. Both versions assume that the analog signal has already been quantized into a thermometer code. A physical analogy is shown in Figure 3-2 where a container filled with an amount of water (charge), proportional to the signal voltage  $S$ , and whose contents are poured into a tray of quantized bins. When a bin is filled with water, the water flows over the top into the next bin. The volume of water is divided between a number of discrete bins.



77-0545-V-16

Figure 3-2: Flow Analogy to Quantization

One version of the sorter takes the charge,  $q$ , residing in each bin and receives them in parallel as in Figure 3-3.

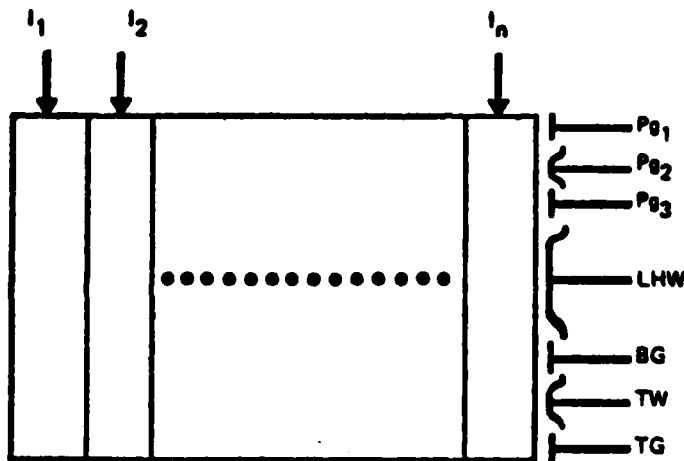


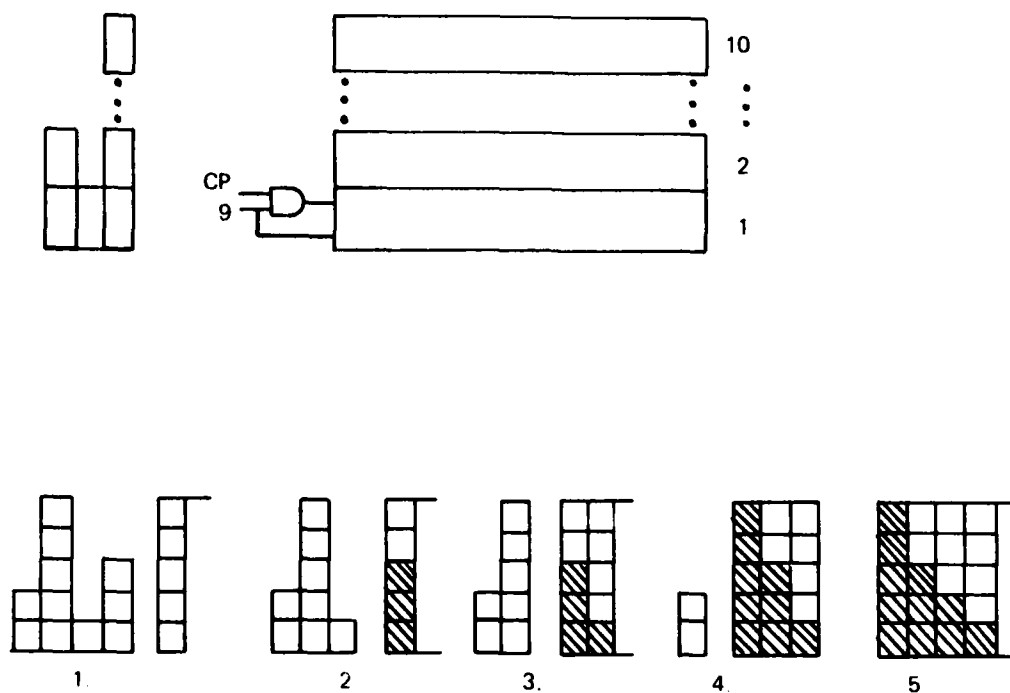
Figure 3-3: CCD Sorter

Thus, there is  $q$  amount of charge shifted from bin  $b_1$  to channel  $I_1$ ,  $q$  amount of charge shifted from bin  $b_2$  to channel  $I_2$  and so on. By mean of gates  $pg_1$ ,  $pg_2$ , and  $pg_3$ , the contents of channels  $I_1$  through  $I_n$  are shifted in parallel to the large holding well LHW. The large holding well is partitioned into  $N$  channels also. Consider a numerical example; a sequence of numbers 4, 7, 5 is quantized at  $q = 1$  so that 4, 7, and 5 bins respectively represent each number. Then Figure 3-5 shows the sequence as it goes through the quantizer, the  $b_1$ ,  $b_2$ , ...  $b_n$  bins, the  $I_1$ ,  $I_2$ , ...  $I_n$  channels and the large holding well. It also shows the removal sequence from the large holding well and the remainder at each stage. The numbers are removed in order of decreasing magnitude 7, 5, 4 which shows the numbers have been sorted by magnitude.

	1	2	3	4	5	6	7	8	9 .....	n
$g_1$ Wells										
5	q	q	q	q	q					
7	q	q	q	q	q	q	q			
4	q	q	q	q						
$I_1$ Channels										
5	q	q	q	q	q					
7	q	q	q	q	q	q	q			
4	q	q	q	q						
Large Holding Well										
4	q	q	q	q						
4,7	2q	2q	2q	2q	q	q	q			
4,7,5	3q	3q	3q	3q	2q	q	q			
First Removal										
	q	q	q	q	q	q	q			
Remainder										
	2q	2q	2q	2q	q					
Second Removal										
	q	q	q	q	q					
Remainder										
	q	q	q	q						
Third Removal										
	q	q	q	q						
Remainder										

Figure 3-4: Sorting Sequence

The other version of the sorter is shown in Figure 3-5.



77-0942-V 17

Figure 3-5. Sorter Fabrication

Here the sorter consists of individually controlled CCD shift registers. Each register is enabled by a clock pulse (CP) and the presence of a charge quantum, . The lower portion of Figure 3-5 shows the numbers 3, 1, 5 and 2 entering the sorter.

At stage 2, the first register will shift only. At stage 3, all five registers will shift and note that the data is then arranged in descending order.

We had been using surface channel CCD's to produce image primitives such as produced by the Median Filter and Non-Maximum Suppression Operators because of wider dynamic range and lower data rate requirements. However, we noticed that the Connected Components Algorithm could also be implemented with buried channel CCD devices so we amended the chip feasibility program to include buried channel devices also. The second version of the sorter was done with surface channel CCD's and the first version was in buried channel. The buried channel device was achieved by ion implantation in the surface channel structure. Probe tests showed that the yields were not high enough to continue processing the buried channel wafers, so they were dropped. The surface channel devices will be used in the demonstration. We could not meet the time schedule if we had started the buried channel device from the ground up as was done successfully on other programs. For the purposes of this project, amending the lower risk surface channel masks seemed to be a reasonable way to produce both surface and buried channel sorters.

Figure 3-6 shows a wafer of the surface channel devices. The mechanical assembly will be available at the DARPA meeting in mid October, with the demonstration scheduled for that meeting. One portion of the demonstration unit is shown in Figure 3-7 with the shift registers mounted in place. The ten shift registers are seen at the top of the unit and ten thumbwheel switches are shown below.

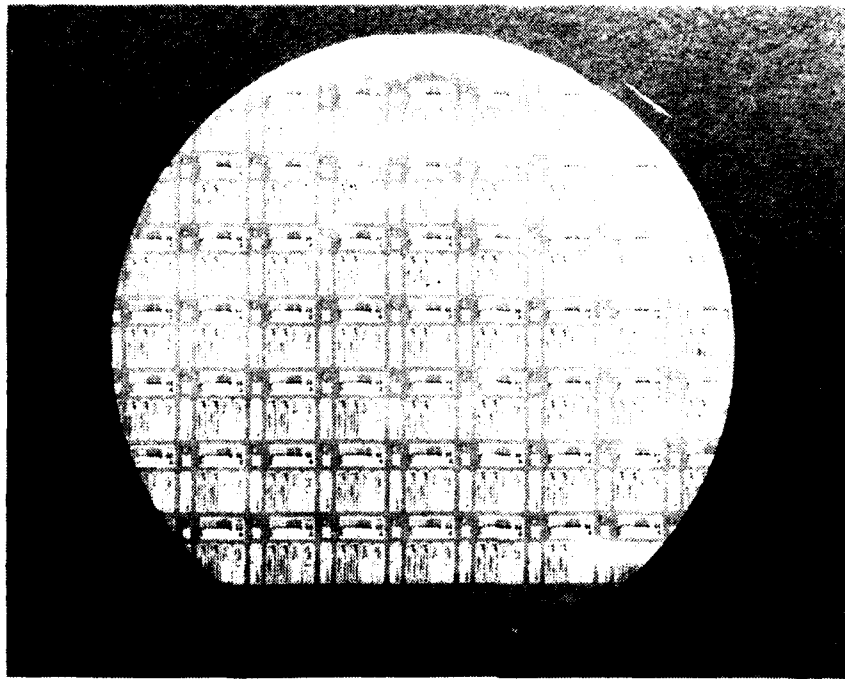


Figure 3-6: Sorter Wafer

These thumbwheels represent the unsorted numbers which the sorter must rearrange in descending order. The observer may dial in any arrangement of numbers which he wishes. The outputs and inputs, i.e., the unsorted and sorted arrangements will be shown on a two-trace oscilloscope.

Westinghouse IR&D accounted for 70 cents of every dollar spent on the DARPA Smart Sensor Project.

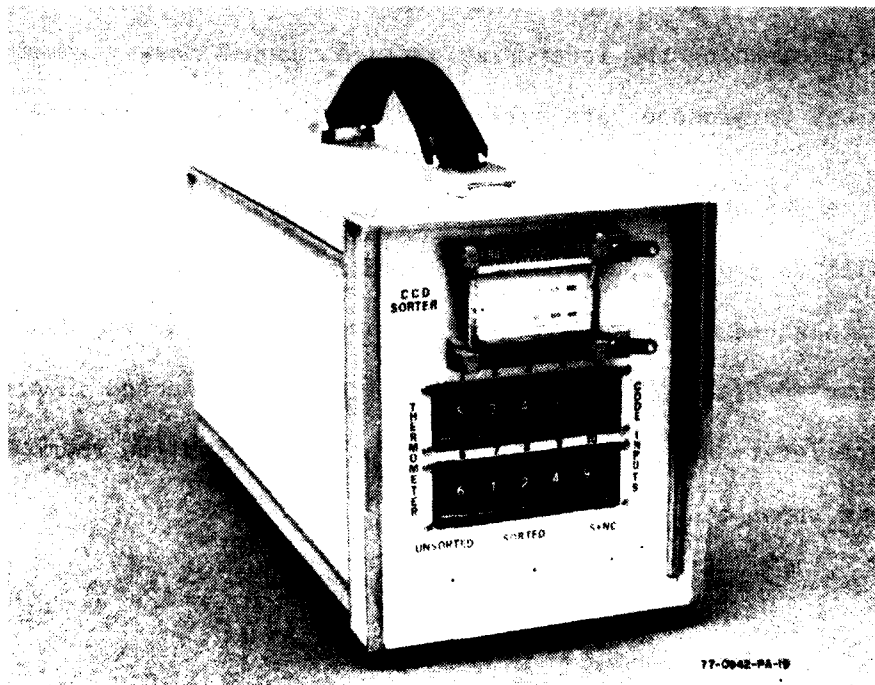


Figure 3-7. Demonstration Unit

**END**

**FILMED**

**2-83**

**DTIC**